ELSEVIER

Discrete Applied Mathematics 117 (2002) 109–131

DISCRETE APPLIED MATHEMATICS

Clique tree generalization and new subclasses of chordal graphs

P. Sreenivasa Kumar[a,*], C.E. Veni Madhavan[b]

[a] *Department of Computer Science and Engineering, Indian Institute of Technology Madras, Chennai 600 036, India*

[b] *Deparment of Computer Science and Automation, Indian Institute of Science, Bangalore 560 012, India*

Received 9 July 1998; revised 14 August 2000; accepted 28 August 2000

Abstract

The notion of a clique tree plays a central role in obtaining an intersection graph representation of a chordal graph. In this paper, we introduce a new structure called the *reduced clique hypergraph* of a chordal graph. Unlike a clique tree, the reduced clique hypergraph is a unique structure associated with a chordal graph. We show that all clique trees of a chordal graph can be obtained from the reduced clique hypergraph; thus the reduced clique hypergraph can be thought of as a generalization of the notion of a clique tree. We then link the *reduced clique hypergraph* notion to minimal vertex separators of chordal graph by proving a structure theorem which shows that the edges of the reduced clique hypergraph are in one–one correspondence with the minimal vertex separators. We also show that a closed-form formula for the number of clique trees of a chordal graph can be derived using these results. Using an algorithmic characterization of minimal vertex separators, we obtain efficient algorithms to compute the reduced clique hypergraph and to count the number of clique trees of a chordal graph. Finally, guided by the *reduced clique hypergraph* structure we propose a few new subclasses of chordal graphs and relate them to each other and the existing subclasses.

Keywords: Chordal graphs; Clique trees; Minimal vertex separators; Counting clique trees

1. Introduction

Chordal graphs form an important subclass of perfect graphs and arise in many practical situations. The notion of clique trees is central to the study of intersection graph representations of a chordal graph [8]. Clique trees are useful in designing efficient sequential and parallel algorithms on chordal graphs. One of the first efficient parallel algorithms to recognize chordal graphs makes use of clique trees [11]. Clique trees have also been used in developing algorithms to solve domination problems on

* Corresponding author. Tel.: +91-44-445-8345; fax: +91-44-235-0509.
E-mail address: psk@iitm.ac.in (P.S. Kumar).

0166-218X/02/$ - see front matter
PII: S0166-218X(00)00336-X

directed path graphs [2]. They also arise in the context of relational databases [1]. A clique tree of a chordal graph essentially represents its tree-like structure.

A chordal graph can have many clique trees, and any particular clique tree does not reflect the complete information about the interactions of the maximal cliques. The aim of the results presented in this paper is to obtain a unique structure that reflects the interactions between the maximal cliques of a chordal graph. With this motivation, we introduce the notion of a *reduced clique hypergraph* (*rch*) of a chordal graph. We show that the notion of *rch* is a generalization of the notion of a clique tree, by proving that all clique trees can be generated from the *rch*. The notion of *rch* yields a lot of insight into the structure of subclasses of chordal graphs.

Linking the notion of *rch* with minimal vertex separators, we prove a structure theorem which characterizes the hyperedges of the *rch* in terms of these separators. Further, we derive a formula for counting the number of clique trees of a chordal graph. We show that using an algorithmic characterization of minimal vertex separators given in [14,15], this formula can be evaluated efficiently, improving upon an existing bound of $O(|V|^{3.5})$ [7].

In contrast to the existing approaches [7,9,13] to the study of clique trees of a chordal graph, the emphasis of our approach is on the development of a unique structure that completely reflects the clique structure of chordal graphs.

We next introduce a few new subclasses of chordal graphs that arise naturally in the framework of the paper. Uniquely representable chordal (*ur*-chordal) graphs, *k*-separator chordal (*k*-sep-chordal) graphs and *arch*-chordal graphs are defined and characterized. The new subclasses can be used to obtain better insight on the complexity of problems that are hard for chordal graphs. We have obtained a few results of this nature by showing that the Hamiltonian circuit problem on 2-sep-chordal graphs and the isomorphism problem on Hamiltonian 2-sep-chordal graphs have linear-time solutions [16,18]. These results generalize known results on maximal outer planar graphs which form a subclass of 2-sep-chordal graphs.

The paper is organized as follows: In Section 2 we collect together the relevant definitions and characterizations regarding chordal graphs. In Section 3 we define clique hypergraph and the reduced clique hypergraph notions. Section 4 contains the characterization of the edges of the *rch* structure in terms of minimal vertex separators. In Section 5, we show that the *rch* can be used to generate all the clique trees. In the final section, we propose new subclasses of chordal graphs. The appendix contains the derivation of the formula for counting the number of clique trees and how it can be evaluated.

2. Preliminaries

An undirected graph $G=(V,E)$ is *chordal* if every cycle of length at least four has a *chord*, i.e., an edge between two nonconsecutive vertices of the cycle. Chordality is a hereditary property, that is, every induced subgraph of a chordal graph is also

chordal. A subset $S \subset V$ is called a *u–v separator* of G if in $G - S$, the vertices u and v are in two different connected components. A *u–v* separator is called a *minimal u–v separator* if no proper subset of it is a *u–v* separator. A *minimal vertex separator* is a minimal *u–v* separator for some u and v. One of the many characterizations of chordal graphs is in terms of minimal vertex separators.

Theorem 1 (Dirac [4]). *An undirected graph is chordal if and only if every minimal vertex separator of it induces a clique.*

A vertex v of a graph G is called *simplicial* if its adjacency set $adj(v)$ induces a clique. An *elimination ordering* σ of a graph G is a bijection $\sigma : \{1,2,\ldots,n\} \to V$, where $|V| = n$. Accordingly, $\sigma(i)$ is the ith vertex in the elimination ordering and $\sigma^{-1}(v)$, $v \in V$ gives the position of v in σ. A *perfect elimination ordering* (*peo*) is an elimination ordering $\sigma = (v_1, v_2, \ldots, v_n)$ where v_i $(1 \leqslant i \leqslant n)$ is a simplicial vertex in the subgraph induced by $\{v_i, v_{i+1}, \ldots, v_n\}$. The following characterization of chordal graphs is well known.

Theorem 2 (Fulkerson and Gross [5] and Golumbic [8]). *An undirected graph is chordal if and only if it has a perfect elimination ordering.*

Given a peo σ of a chordal graph G, $N(v,\sigma)$ denotes the set of vertices adjacent to v that appear later than v in σ. That is,

$$N(v,\sigma) = \{x \in adj(v) \colon \sigma^{-1}(x) > \sigma^{-1}(v)\}.$$

The set $N(v,\sigma)$ is called the *monotone adjacency set* of v with respect to σ. The graph G can be constructed by starting with an empty graph and adding vertices in the order $\sigma(n), \sigma(n-1), \ldots, \sigma(1)$, making each added vertex v adjacent to all the vertices in $N(v,\sigma)$. We call this process the *reconstruction* of G with respect to σ. Throughout the paper, we use the following notation: $V_i(\sigma) = \{v : \sigma^{-1}(v) \geqslant i\}$. $G_i(\sigma)$ is the subgraph of G induced by $V_i(\sigma)$. When the peo σ is clear from the context, we use V_i and G_i to mean $V_i(\sigma)$ and $G_i(\sigma)$, respectively. We call a chordal graph *nontrivial* if it has at least two maximal cliques. In this paper we assume that the chordal graphs under discussion are nontrivial, unless otherwise stated.

Chordal graphs can be recognized in linear time. Maximum cardinality search (MCS) [17] and lexicographic breadth first search (LBFS) [8] are two well-known algorithms that generate an elimination ordering of a graph which would be perfect if and only if the graph is chordal.

A separator is *minimal* if it contains no other separator. Note that a minimal vertex separator is not necessarily a minimal separator, as the minimal *u–v* separator may contain the minimal *x–y* separator for some other pair of vertices x, y. The minimal separators of a chordal graph have the following interesting property which can be established easily.

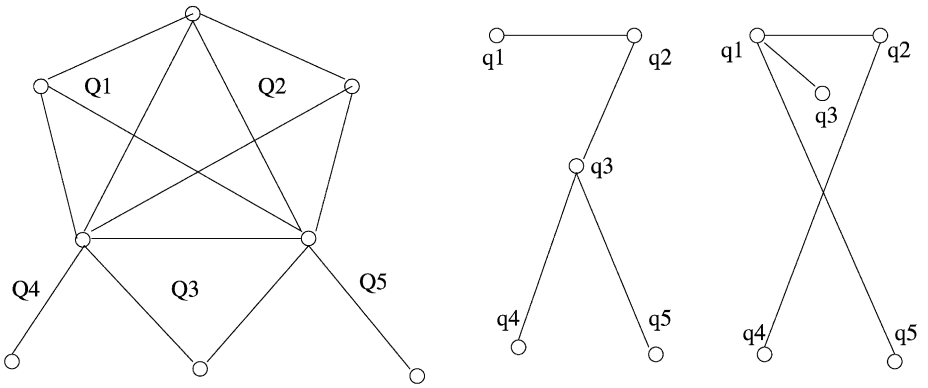

Fig. 1. A chordal graph and two of its clique trees.

Lemma 1. *If S is a minimal separator of a chordal graph G, then each of the connected components of $G - S$ has a vertex that is adjacent to all the vertices of S.*

Corollary 1. *If S is a minimal u–v separator of a chordal graph G, then the connected components of $G - S$ that contain either u or v have a vertex that is adjacent to all the vertices of S.*

Chordal graphs have an interesting intersection graph representation. Buneman [3] and Gavril [6] have shown that an undirected graph is chordal if and only if it is an intersection graph of a family of subtrees of a tree. A *tree representation* of a chordal graph G is a pair $(T, \mathscr{F})$ where T is a tree and $\mathscr{F}$ is a family of subtrees of T such that the intersection graph of $\mathscr{F}$ is isomorphic to G. Gavril [6] has further shown that given a chordal graph G, it is possible to construct a tree T with vertex set $K = \{q_1, q_2, \ldots, q_r\}$ where q_i corresponds to the maximal clique Q_i of G, such that $(T, \{R_{v_1}, Rv_2, \ldots, R_{v_n}\})$ is a tree representation of G. Here, each R_{v_i}, $1 \leqslant i \leqslant n$, is the set of maximal cliques that contain the vertex v_i, i.e., $R_{v_i} = \{q_j : v_i \in Q_j\}$. Such a tree representation of G is called a *clique tree* of G. The sets R_i determine the edges of T, not necessarily in a unique manner. An undirected graph is chordal if and only if it has a clique tree. Fig. 1 shows a chordal graph and two of its clique trees.

Theorem 3 (Gavril [6] and Buneman [3])**.** *The following propositions are equivalent*:

1. *G is a chordal graph.*
2. *G is the intersection graph of a family of subtrees of a tree.*
3. *There exists a tree T with vertex set $\{q_1, q_2, \ldots, q_r\}$ such that for each vertex v, the set $R_v = \{q_i : v \in Q_i\}$ induces a subtree of T. Here q_i represents the maximal clique Q_i.*

The class of k-trees is a proper subclass of chordal graphs defined recursively as follows: A k-clique is a k-tree and if H is a k-tree with a k-clique Q then adding

a vertex v to H and making it adjacent to all the vertices in Q gives another k-tree. k-trees generalize the class of trees which are nothing but 1-trees.

3. Clique hypergraphs

The definition of a clique tree of a chordal graph states that a tree T with the set of vertices $K = \{q_1, q_2, \ldots, q_r\}$ is a clique tree if the sets $R_v = \{q_i : v \in Q_i\}$ induce connected subtrees in T for all v. Each R_v imposes a restriction on how the vertices of T can get connected to each other. To formalize this, we define the *clique hypergraph* $H(G)$ of a chordal graph G as $H(G) = (K, \{R_{v_1}, R_{v_2}, \ldots, R_{v_n}\})$. A tree T with vertex set K is said to *satisfy* an edge R_{v_i} of $H(G)$ if R_{v_i} induces a connected subtree in T. T is said to *satisfy* $H(G)$ if it satisfies all the edges of $H(G)$. Clearly T is a clique tree of G if and only if it satisfies $H(G)$. We call the edges of $H(G)$ the *initial restriction sets* of G. The information in $H(G)$ about the clique trees can be refined so that $H(G)$ becomes a minimal structure in some sense. Towards this end, we define two operations on $H(G)$. The hypergraph obtained by repeatedly performing these operations on $H(G)$ until they can no longer be performed is called the *reduced clique hypergraph* (*rch*) of G, denoted by $H'(G)$. We formulate these operations by means of the following lemmas.

A subset $R \subset K$ of the vertices of the clique hypergraph is called a *restriction set* if either it is one of the initial restriction sets or it is the intersection of two or more restriction sets. A *nontrivial* restriction set is one that has at least two elements in it. Henceforth, while dealing with clique hypergraphs we ignore the trivial restriction sets. We now show that the restriction sets behave similar to the initial restriction sets.

Lemma 2. *If $R = \{q_1, q_2, \ldots, q_s\}$ is a restriction set of a chordal graph G then R induces a connected subgraph in any clique tree of G.*

Proof. By the definition of the restriction set, we know that there exist initial restriction sets $R_1, R_2, \ldots, R_p$ such that $R = \bigcap_{i=1}^{p} R_i$. For $p = 1$, the lemma is trivially true. So, let $p \geqslant 2$. Let T' be the subtree induced by the set R in a clique tree T of G. Suppose T' is not connected. Let T_1, T_2 be any two connected components of T'. Choose two vertices $y \in T_1$ and $z \in T_2$ such that the vertices x_i in the unique path $(y, x_1, x_2, \ldots, x_j, z)$ connecting y and z in T are not in T'. It is easily seen that such a choice can always be made. Since $y, z \in R_i$, $i = 1, \ldots, p$ and since each R_i induces a connected subgraph in T, we have $x_1, x_2, \ldots, x_j \in R_i$, $i = 1, \ldots, p$. This implies that $x_1, \ldots, x_j$ belong to R and hence to T'. Thus T' is connected, a contradiction. □

Lemma 3. *A tree T satisfies the hypergraph $H_1 = (K, \{R_1, \ldots, R_n\})$ if and only if it satisfies the hypergraph H_2 obtained from H_1 by incorporating all the nontrivial intersections of the edges of H_1 as additional edges.*

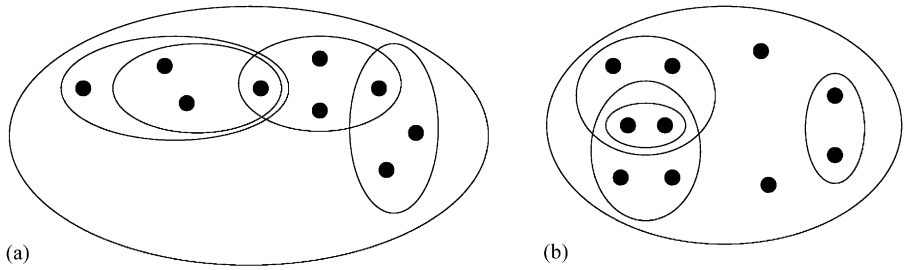

Fig. 2. (a) Reducible hyperedge and (b) irreducible hyperedge.

Proof. (If) Obvious as $R_1,\ldots,R_n$ are also present in H_2.
(Only if) Follows from Lemma 2. □

The above lemma shows that introducing the intersection of the edges of $H(G)$ as additional edges in $H(G)$ refines the information represented by $H(G)$ about the clique trees of G.

We call an edge R of $H(G)$ a *reducible* edge if there exists a set $S_1,S_2,\ldots,S_p$ of edges of $H(G)$ such that (i) $S_i \subset R$ for all i, (ii) every element of R is in some S_i and (iii) the partial hypergraph $(R,\{S_1,S_2,\ldots,S_p\})$ is connected. Otherwise, R is called an *irreducible* edge. Fig. 2 shows an example of a reducible edge.

A reducible hyperedge R induces a connected subgraph if all of its subhyperedges $S_1,\ldots,S_p$ induce connected subgraphs. Conversely, the only way a tree T can satisfy R, is that T satisfies each of the subhyperedges $S_1,\ldots,S_p$. Thus, we have

Lemma 4. *A tree T with vertex set K satisfies a hypergraph $H_1=(K,\{R_1,\ldots R_r\})$ if and only if it satisfies H_2 obtained from H_1 by removing all the reducible hyperedges.*

Proof. (If) Suppose T satisfies H_2. In addition to the edges in H_2, H_1 has some reducible edges. From the definition of reducible edges, it is easy to show that corresponding to each such edge R there is a collection of edges $R'=\{S_1,S_2,\ldots,S_p\}$ such that each S_i is irreducible and hence in H_2. These edges together contain all the vertices in R. Thus the only way T can satisfy R is by satisfiying the collection R'. Again, by the definition of reducible edges, R' is a connected subhypergraph. So if all the edges are individually satisfied by T, T would satisfy R. Thus if T satisfies H_2, then it satisfies H_1.
(Only if) Obvious. □

Now, we define the *reduced clique hypergraph* (*rch*) of a chordal graph. The hypergraph $H'(G)$ obtained from $H(G)$ by (i) repeatedly introducing the intersections of existing hyperedges of $H(G)$ as additional edges until all such intersections are

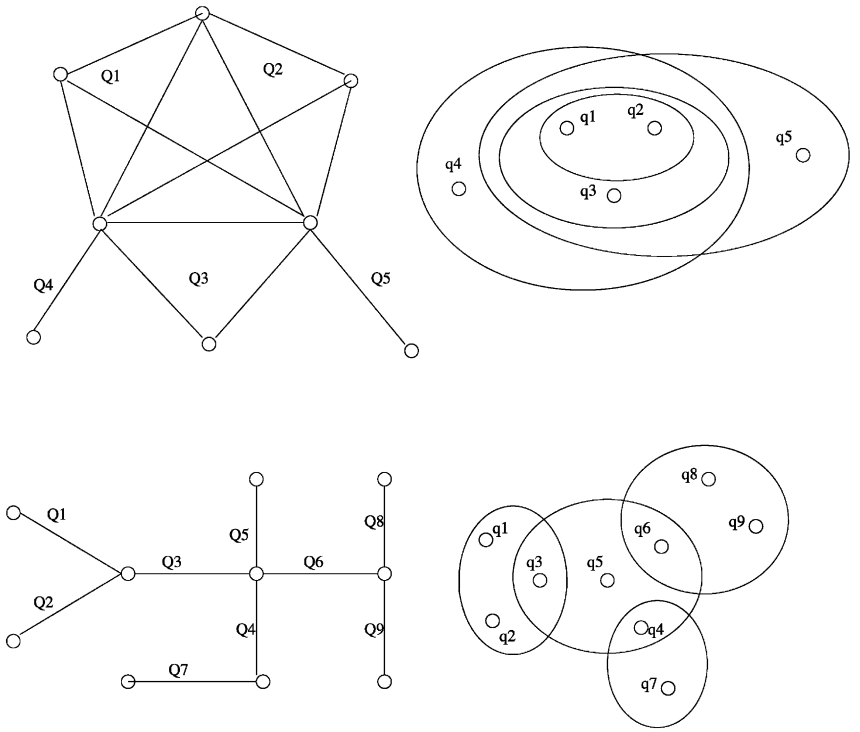

Fig. 3. Chordal graphs and their *rch*.

introduced and (ii) removing all the reducible edges from the resulting $H(G)$, is called the *reduced clique hypergraph* of G.

Theorem 4. *A tree T is a clique tree of the chordal graph G if and only if it satisfies the reduced clique hypergraph $H'(G)$.*

Proof. Follows from Lemmas 3 and 4. □

Fig. 3 shows the *rch* of some chordal graphs. On the left side are two graphs with their reduced clique hypergraphs shown on the right side. In the graphs, maximal cliques are indicated as $Q_1, Q_2, \ldots$. The vertices in the *rch*'s corresponding to these maximal cliques are shown as $q_1, q_2, \ldots$. It can be verified that the edges in the *rch*'s are obtained using the steps given in the definition of *rch.*

4. Characterizing the edges of the reduced clique hypergraph

Consider the *rch* of a tree G. As any two vertices u, v of G can be present in at most one maximal clique (an edge) of G, $|R_u \cap R_v| \leqslant 1$. Therefore, the reduced clique

hypergraph of G will be the same as its clique hypergraph. Fig. 3(b) shows the *rch* $H'(G)$ of a tree. Since R_x, where x is a leaf of G, has only one element, the nontrivial edges of $H'(G)$ are contributed by the internal vertices of G. We note that there is a one–one correspondence between the internal vertices of G and the hyperedges of $H'(G)$. The edge corresponding to a vertex v is the set of all maximal cliques (i.e., edges) of G that contain v. We propose to generalize this correspondence to chordal graphs. We discover that the set of minimal vertex separators of a chordal graph play the role of internal vertices.

An algorithmic characterization of minimal vertex separators in terms of monotone adjacency sets of MCS peo's is proposed in [14,15]. We provide the definitions and main results from the above paper that are required for the proof presented in this section.

A set $B \subset V(G)$ is defined to be a *base set* of a chordal graph G with respect to a peo σ if there exists a vertex v with $\sigma^{-1}(v) = t$ such that (i) $B = N(v, \sigma)$ and (ii) B is not a maximal clique in $G_{t+1}(\sigma)$. For a base set B, the vertices which satisfy (i) and (ii) above are called its *dependent vertices* with respect to σ. We denote this set of vertices by $D(B, \sigma)$. The size of the set $D(B, \sigma)$ is called the *multiplicity* of the base set B and is denoted $\mu(B, \sigma)$.

Though the set of base sets is defined with respect to a specific peo, it can be shown that irrespective of the peo used, we obtain the same set of base sets and they, in fact, constitute the set of all minimal vertex separators [15].

Theorem 5. *Let α and β be two peo's of a chordal graph G.* (i) *If B is a base set of G with respect to α then it is also a base set of G with respect to β, and* (ii) $\mu(B, \alpha) = \mu(B, \beta)$.

Theorem 6. *A subset $S \subset V(G)$ is a minimal vertex separator of a chordal graph G if and only if S is a base set of G.*

We shall henceforth use the terms *base sets* and minimal vertex separators interchangeably. A useful characterization of base sets is as follows:

Theorem 7. *A clique S of a chordal graph G is a base set with multiplicity r if and only if there exists $(r+1)$ maximal cliques $Q_0, Q_1, \ldots, Q_r$ such that $Q_i \cap Q_j = S$ and $Q_i - S$, $Q_j - S$ are in different components of $G - S$ for all $0 \leqslant i < j \leqslant r$.*

In this section we characterize the edges of the reduced clique hypergraph $H'(G)$ of a chordal graph G in terms of its base sets. We use induction on the number of base sets to establish the theorem. In the induction step, we decompose G into smaller chordal graphs by removing a minimal base set.

In the following lemma, we characterize the minimal separators of a chordal graph.

Lemma 5. *A clique S is a minimal separator of a chordal graph G if and only if S is a minimal base set of G.*

Proof. (If) Suppose $S_1 \subset S$ is also a separator of G and further let S_1 be a minimal separator contained in S. Consider the connected components of $G - S_1$. By Lemma 1, each of them must contain at least one vertex that is adjacent to all the vertices of S_1. Now, it is easy to find two maximal cliques Q_1 and Q_2 such that $Q_1 \cap Q_2 = S_1$ and $Q_1 - S_1$ and $Q_2 - S_1$ are in different components of $G - S_1$. By Theorem 7, S_1 is also a base set of G, contradicting the minimality of base set S. Thus S is a minimal separator of G.

(Only if) Let S be a minimal separator of G. As G is chordal, S is a clique. By an argument similar to the one used in the "if" part of the proof it follows that S is base set of G. Since no vertex separator of G is properly contained in S, there can be no base set properly contained in S. Thus S is a minimal base set of G. □

For a base set B, let $r(B)$ denote the number of connected components of $G(V - B)$. The *decomposition* $G[B]$ of G with respect to B is obtained as follows: Let M_i', $1 \leqslant i \leqslant r(B)$ be the connected components of $G(V - B)$. Obtain M_i from M_i' by adding to it those vertices of B that are adjacent to some vertices in M_i' along with the edges. $G[B]$ is the graph consisting of the components M_i. Fig. 4 illustrates this definition.

For a base set B, let $d(B)$ denote the number of connected components of $G - B$ that contain at least one vertex which is adjacent to all the vertices of B. These are called the *dependent* components of B. Also, $d(B) = \mu(B) + 1$. The following lemma relates $r(B)$ and $d(B)$.

Lemma 6. *Let B be a base set of a chordal graph G. Then* (i) $r(B) \geqslant d(B)$ *and* (ii) $r(B) = d(B)$ *if and only if B is a minimal base set of G.*

Proof. (i) Obvious.

(ii) (If) Assume that $r(B) > d(B)$. As B has only $d(B)$ dependent components, $G[B]$ has at least one component say M_1 such that $M_1 \cap B \subset B$. Let M_2 be a component of $G[B]$ that contains a dependent vertex of B with respect to σ. Then $M_1 \cap M_2 = B' \subset B$. Now B' is a clique separator that is contained in two maximal cliques say Q_1, Q_2 such that $Q_1 \subseteq M_1, Q_2 \subseteq M_2$. Further $Q_1 - B', Q_2 - B'$ are in different components of $G - B'$. Thus by Theorem 7, B' is also a base set of G, contradicting the minimality of B. Thus $r(B) = d(B)$.

(Only if) If $r(B) = d(B)$ then clearly, B is a minimal separator of G. By Lemma 5, B is a minimal base set of G. □

For a clique C of G, we denote by $R(C)$ the set of maximal cliques that contain C, i.e., $R(C) = \{q_j : C \subset Q_j\}$. Note that $R(\{v\}) = R_v$, defined earlier. For a hyperedge E, let H_E denote the subhypergraph generated by the elements of E. By *components* of E, we mean the connected components of H_E, obtained after deleting the edge E and reducible

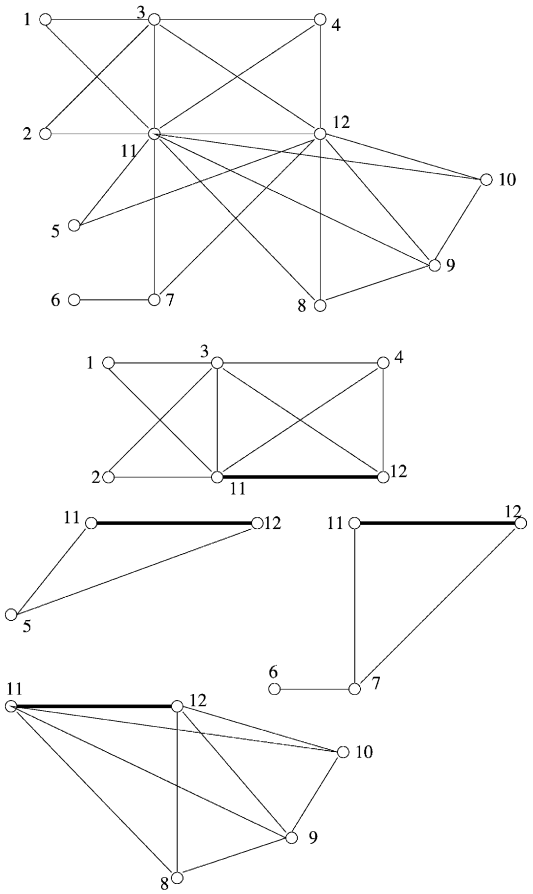

Fig. 4. The decomposition of a chordal graph G with respect to a base set B, the base set chosen is the edge (11,12).

edges if any. In the following theorem we show that the reduced clique hypergraph of a chordal graph is precisely the hypergraph $(K=\{q_1,q_2,\ldots,q_m\},R(B_1),R(B_2),\ldots,R(B_r))$ where B_i are the base sets of G.

Lemma 7. *In a chordal graph G, if C is a clique that is not a subset of a base set of G, then C is contained in exactly one maximal clique of G.*

Proof. We use induction on the number of base sets of G. Suppose the lemma holds for a chordal graph that has t or less number of base sets. Assume that G has $t+1$ base sets. Choose a minimal base set B of G and consider the decomposition $G[B]$ of G with respect to B. Since by assumption C is not a subset of B, C is contained in exactly one of the components of $G[B]$. Thus, by induction C is contained in exactly one maximal clique of G. □

Theorem 8. *The edges of the reduced clique hypergraph $H'(G)$ of a chordal graph G are in one–one correspondence with the base sets of G. The edge corresponding to a base set B is $R(B)$. The number of components of $R(B)$ is $d(B)$.*

Proof. We use induction on the number of base sets of G. The induction hypothesis is as follows:

(i) $H'(G)=(\{q_1,q_2,\ldots,q_m\},R(B_1),R(B_2),\ldots,R(B_r))$ where $B_1,B_2,\ldots,B_r$ are the base sets of G,
(ii) $R(B_i)$ has $d(B_i)$ components, and
(iii) If C is any clique of G, $H_{R(C)}$ is connected.

Basis case: Suppose G has exactly one base set B of multiplicity t. Let $R(B)=\{q_0,q_1,\ldots,q_t\}$. Clearly, $Q_0,\ldots,Q_t$ are the only maximal cliques of G. $R(B)$ has $d(B)=\mu(B)+1=t+1$ components, each component being a single vertex. Also, $R(B)=\bigcap_{v\in B}R_v$. For any vertex $v\notin B$, by Lemma 7, R_v is a trivial restriction set containing a single element. Therefore, $R(B)$ is the only edge of $H'(G)$. Suppose C is any clique of G. If C is contained in B, then $R(C)=R(B)$ which is a hyperedge of $H'(G)$. Otherwise, by Lemma 7, C is contained in a unique maximal clique say q_i and $R(C)=\{q_i\}$. In either case, the induction hypothesis is satisfied.

Now, assume that the induction hypothesis is true for any chordal graph that has at most $j-1$ base sets. Let G be a chordal graph with j base sets. Let B be a minimal base set of G. Consider the decomposition $G[B]$ of G with respect to B. As B is minimal $G[B]$ has $d(B)$ components say $M_1,M_2,\ldots,M_{d(B)}$, each of which is a chordal graph with at most $j-1$ base sets. Let H_i denote the reduced clique hypergraph of M_i. We will construct a hypergraph H from the H_i's and show that H is the reduced clique hypergraph of G. Clearly, $R(B)=R^1(B)\cup R^2(B)\cup\cdots\cup R^{d(B)}(B)$, where $R^i(B)$ is the set of maximal cliques of M_i that contain B, i.e., $R^i(B)=\{q_k\colon Q_k\in M_i, B\subset Q_k\}$. By the induction hypothesis, $R^i(B)$ generates a connected subhypergraph in H_i. Now, construct H as follows: Let H consist of all the edges of the H_i's. Further, add a hyperedge that intersects with each H_i in exactly $R^i(B)$. Thus the new hyperedge is $R(B)$. Also, it has $d(B)$ components. We show that H is the reduced clique hypergraph of G. We will first show that condition (iii) of the induction hypothesis holds.

We need to show that for any clique C of G, the subhypergraph of H generated by the set $R(C)$ is a connected hypergraph. As B is a separator, C is either contained in exactly one of the M_i's or $C\subseteq B$ and hence it is contained in all the M_i's. If C is contained in any M_i, then by the induction hypothesis, $R(C)$ generates a connected hypergraph. Suppose $C\subset B$. Then $R(C)=R^1(C)\cup R^2(C)\cup\cdots\cup R^{d(B)}(C)$ where $R^j(C)$ is the set of maximal cliques of M_j that contain C. Again by the induction hypothesis, each $R^j(C)$ generates a connected hypergraph. Further since $C\subset B$, $R(B)\subset R(C)$. Note that $R(B)$ is an edge of the hypergraph H, and $R(B)$ intersects the reduced clique hypergraphs of all M_i. Therefore the presence of $R(B)$ in H assures that $R(C)$ generates a connected subhypergraph of H.

Now, we show that H is the reduced clique hypergraph of G. By definition, H includes all the hyperedges of all H_i's and in addition the edge $R(B)$. We show that $R(B)$ is the only additional edge that needs to be included, thus proving that H is the reduced clique hypergraph of G.

Consider any subset S of $V(M_i)$, the vertex set of M_i. Let $R=\bigcap_{v\in S} R_v$. In H_i, either R is a hyperedge or it generates a connected subhypergraph depending on whether it is irreducible or otherwise. Now, when G is obtained by taking the union of M_i, the initial restriction sets associated with some of the vertices of S get affected as they may now be contained in additional maximal cliques. But unless all of them are contained in an additional maximal clique, R will not change. Thus R gets affected if and only if $S=B$. Thus, the only additional edge that needs to be included is $\bigcap_{v\in B} R_v$ which is precisely $R(B)$. Since $R(B)$ intersects with any subhypergraph H_i in exactly one connected component, $R(B)$ is an irreducible edge. Thus H is the reduced clique hypergraph of G. □

The proof of the above theorem is illustrated in Fig. 5.

Corollary 2. *Minimal base sets of a chordal graph G correspond to maximal hyperedges of its reduced clique hypergraph $H'(G)$ and similarly maximal base sets correspond to minimal hyperedges.*

Corollary 3. *If B is a maximal base set of a chordal graph G then B is contained in exactly $\mu(B)+1$ maximal cliques of G.*

5. Enumerating clique trees

In this section, we first discuss the details of certain characterizations of clique trees of chordal graphs. We then demonstrate that the notion of *rch* is a generalization of the notion of a clique tree by showing that all clique trees can be generated from the *rch*.

Clique trees of a chordal graph have been characterized by Bernstein and Goodman [1] in an interesting way. The intersection graph of the set of maximal cliques of an undirected graph G is called its *clique graph* and is denoted by $q(G)$. Associate integer weights to the edges of the clique graph, the weight of an edge being the size of the intersection of the maximal cliques corresponding to its endpoints. In [1] (see also [7]), it is proved that the clique trees of a chordal graph G are exactly the maximum weighted spanning trees of the weighted clique graph of G. In [7], Gavril proposed an algorithm to generate the maximum spanning trees of a weighted undirected graph. Using this approach, counting the number of clique trees of a chordal graph takes $O(|V|^{3.5})$.

A characterization of the clique trees of a chordal graph similar to that of [1] is proposed in [13]. The t-overlap clique graph $q_t(G)$ of a graph G is a graph with

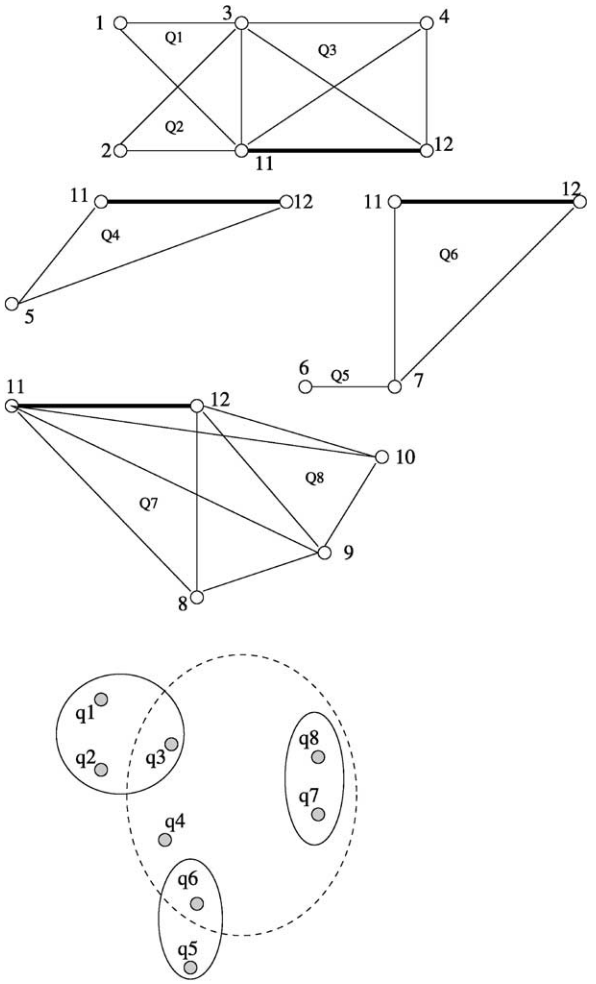

Fig. 5. Illustration of the proof of theorem on *rch* edges.

vertices corresponding to the maximal cliques of G and two vertices are adjacent if and only if the corresponding maximal cliques intersect in at least t elements. Shibata proposed the following algorithm to obtain a clique tree of a chordal graph G: Start with $q_s(G)$ where s is the smallest integer such that $q_s(G)$ is an edgeless graph. Set $T_s = q_s(G)$. To get T_{i-1} from T_i, throw in as many edges of weight $(i-1)$ as possible such that the resulting T_{i-1} remains a forest. T_1 is a clique tree of G.

We now propose a new characterization of the clique trees of a chordal graph by giving a recursive algorithm to generate the clique trees of a chordal graph assuming that we are given its minimal vertex separators. This also demonstrates that the notion of *rch* is a generalization of the notion of a clique tree.

```
Algorithm Generate-Clique-Trees (G);
begin
    if G is not a clique then
    begin
        choose a minimal base set B_i of G;
        determine the components M_1, M_2, ..., M_{μ(B_i)+1} of G[B_i];
        for s = 1 to μ(B_i) + 1 do
            T_s ← Generate-Clique-Trees(M_s);
        let T be the forest defined by T_1 ∪ T_2 ∪ ··· ∪ T_{μ(B_i)+1};
        choose an arbitrary spanning tree T' of the complete graph on
        {t_1, t_2, ..., t_{μ(B_i)+1}}, where t_i represents T_i;
        for each edge (t_j, t_k) of T' do
        begin
            choose a node x_j of T_j such that B_i is a
            subset of the maximal clique corresponding to x_j;
            similarly, choose a node x_k of T_k such that B_i is a
            subset of the maximal clique corresponding to x_k;
            add the edge (x_j, x_k) to T
        end
    end
    else
    begin
        let G be the maximal clique Q_i;
        T ← a single node graph consisting of q_i;
    end;
end;
```

Theorem 9. *A tree is a clique tree of a chordal graph if and only if it is generated by Generate-Clique-Tree.*

Proof. (If) Assume that Generate-Clique-Tree works correctly on chordal graphs with t or less number of base sets. Suppose G has $t+1$ base sets. Let T be a tree generated by the algorithm working on G. Then T is obtained from $T_1, T_2, \ldots, T_{\mu(B_i)+1}$ with the help of T'. By induction, T_i's are clique trees of the components of $G[B]$. Thus by Theorem 4, all the *rch* edges other than $R(B)$ are satisfied by T. The choice of T' as a spanning tree connecting the nodes x_i's (as defined in the algorithm), ensures that $R(B)$ is also satisfied. Thus by Theorem 4, T is a clique tree of G.

(Only if) It is enough to show that all possible ways of satisfying the *rch* $H'(G)$ are explored by the algorithm. We use induction on the structure of the *rch* to show this. In the basis case, $H'(G)$ has a single hyperedge R containing isolated nodes only. In terms of graph G, this corresponds to having one base set B contained in all the maximal cliques of G. In this case, all the components of $G[B]$ would be cliques. The algorithm first generates single-node trees for these cliques. It then connects these points

using all possible spanning trees. Thus, it generates all possible ways of satisfying R. For the induction step, consider a maximal hyperedge R that contains components $R^1, R^2, \ldots, R^p$. By induction, all possible ways of satisfying R^i are explored by the algorithm. In terms of the graph G, maximal hyperedge R corresponds to a minimal base set, say, B. Each of the R^i's is part of the *rch* of the components of $G[B]$. The nodes in each R^i correspond to maximal cliques that contain B and hence are all candidates which can be chosen as an x vertex from T_i in the algorithm. All spanning trees that connect x vertices are explored by the algorithm. Thus all possible ways of satisfying R are explored by the algorithm. □

A characterization of the clique trees of a chordal graph similar to the one presented above has been previously developed by Ho and Lee [9]. However, the emphasis of our approach to the study of the clique structure of chordal graphs is on introducing a new structure, namely the *rch*, that captures all the clique interactions in the graph and generalizes the notion of clique trees. As we show in the next section the insight provided by the *rch* leads us to propose several new subclasses of chordal graphs.

In [9], a formula for counting the number of clique trees of a chordal graph is presented. In an appendix to the paper, we show that essentially the same formula can be derived using the results of this paper. As known from [7], the problem of counting the number of clique trees of a chordal graph can be solved in polynomial time. We provide a different algorithm to solve this problem. This solution uses the formula mentioned above and in addition an algorithmic characterization of minimal vertex separators proved in [14,15].

6. New subclasses of chordal graphs

In this section, we introduce certain new subclasses of chordal graphs that emerge naturally in the framework of the paper. We also present results that relate these classes with the existing subclasses. The characterizations of the classes are all in terms of their *rch* structure.

6.1. Uniquely representable chordal graphs

In the context of the study of clique structure of chordal graphs, a natural and interesting question that arises is that of characterizing chordal graphs that have exactly one clique tree. A chordal graph is called a *uniquely representable chordal graph* (briefly *ur-chordal graph*) if it has exactly one clique tree. These graphs are much simpler in their structure and can be an initial subclass on which a problem can be tried before it is attempted for general chordal graphs.

Theorem 10. *A chordal graph is uniquely representable if and only if* (i) *There is no proper containment between any two base sets and* (ii) *All base sets are of multiplicity one.*

Proof. (If) Let G be a chordal graph satisfying conditions (i) and (ii). We show that the *rch* $H'(G)$ of G is a tree, thus showing that G has a unique clique tree. As no two base sets of G are contained in each other, no two hyperedges in $H'(G)$ are contained in each other (Theorem 8, p. 11). It also follows that any two hyperedges of $H'(G)$ intersect in exactly one node, for otherwise, the intersection will be a subedge in the two intersecting hyperedges. Further, since all the base sets are of multiplicity one, all hyperedges have exactly two nodes in them (Theorem 8). Hence, $H'(G)$ is a tree and is the unique clique tree of G.

(Only if) As the Theorem 8 is a characterization, the arguments of the "If" part of the proof can be used in the reverse direction to show that a chordal graph with a unique clique tree satisfies the required conditions on its base sets. □

Theorem 11. *A chordal graph G is uniquely representable if and only if every base set of G is contained in exactly two maximal cliques of G.*

Proof. (If) Assume that every base set of G is contained in exactly two maximal cliques. Suppose B_1 and B_2 are two base sets such that $B_1 \subset B_2$. Thus B_1 is contained in the two maximal cliques that contain B_2. In addition, B_1 is contained in at least one more maximal clique because B_1 has at least one dependent vertex. So, B_1 is contained in three maximal cliques, a contradiction. Thus there is no proper containment between any two base sets.

Also, if a base set B has $\mu(B) \geqslant 2$, then B is contained in at least three maximal cliques. Thus all base sets of G have multiplicity one. By Theorem 10, G has a unique clique tree.

(Only if) A hyperedge of the *rch* $H'(G)$ of a *ur*-chordal graph G is of size two. Hence the base set of G corresponding to the hyperedge is contained in exactly two maximal cliques, by Theorem 8. □

Proper interval graphs or indifference graphs arise in many practical situations [19]. Here, we show that these graphs form a subclass of *ur*-chordal graphs.

Theorem 12. *Proper interval graphs are uniquely representable.*

Proof. Recall that an interval graph is a proper interval graph if and only if it does not contain an induced subgraph that is isomorphic to a claw, i.e., $K_{1,3}$ [8]. Let G be a proper interval graph. Suppose B_1 and B_2 are base sets of G such that $B_1 \subset B_2$. Let σ be a peo of G and let v_1 and v_2 be dependent vertices of B_1 and B_2, respectively, with respect to σ. Let Q be a maximal clique of G that properly contains B_2. Let $v_3 \in Q - B_2$ and let x be any vertex in B_1. Clearly, x is adjacent to v_1, v_2 and v_3. Then, $\{x\}$, $\{v_1, v_2, v_3\}$ form a $K_{1,3}$. Thus no two base sets of G are contained in each other. Suppose there is a base set B with multiplicity two or more. Let v_1, v_2 be two dependent vertices of B with respect to σ. Again let Q be a maximal clique that properly contains B and let $v_3 \in Q - B$. Then, $\{x\}, \{v_1, v_2, v_3\}$ induce a $K_{1,3}$. Thus all base sets of G are of multiplicity one. By Theorem 10, G is a uniquely representable chordal graph. □

However, it is easy to see that not all uniquely representable interval graphs are proper interval graphs.

We can recognize a *ur*-chordal graph by testing whether the *rch* of the given chordal graph is a tree.

6.2. *k*-separator-chordal graphs

The class of k-trees has fixed-size minimal vertex separators, namely k, and the size of the maximum clique size is $k+1$. We can relax the condition on the clique size and obtain a super class of k-trees that has the same *rch* structure of k-trees.

We call a chordal graph a *k-separator-chordal graph* (briefly *k-sep-chordal graph*) if every minimal vertex separator is of size exactly k. k-sep-chordal graphs generalize k-trees. A k-chordal graph is a chordal graph with maximum clique size $(k+1)$. k-chordal graphs and k-sep-chordal graphs are incomparable subclasses of chordal graphs. k-sep-chordal graphs can have cliques of arbitrary large size whereas k-chordal graphs can have minimal vertex separators of arbitrary size less than k. Also, the class of chordal graphs whose base sets are of size less than or equal to k properly contain the class of k-chordal graphs. Thus k-sep-chordal graphs and k-chordal graphs are two different generalizations of the notion of k-trees.

Hamiltonian 2-trees turn out to be maximal outer planar graphs and they can be recognized in linear time. We have generalized this result and shown that Hamiltonian 2-sep-chordal graphs can be recognized in linear time. Also, as with maximal outer planar graphs, the isomorphism problem for Hamiltonian 2-sep-chordal can also be solved in linear time [16,18].

6.3. Acyclic rch chordal graphs

We first show that the structure of *rch* of k-trees is very simple. We then generalize and obtain the largest subclass of chordal graphs that shares this simplicity.

A (Berge) cycle in a hypergraph is a sequence $x_1, E_1, x_2, E_2, \ldots, x_r, E_r, x_1$ where x_i are distinct vertices and E_i are distinct edges of the hypergraph and $x_i \in E_{i-1} \cap E_i,\ 1 < i \leqslant r$ and $x_1 \in E_1 \cap E_r$. We call a hypergraph Berge-acyclic if it does not have cycles.

Lemma 8. *If G is a k-tree then its* rch $H'(G)$ *has the following properties*: (i) *No hyperedge is properly contained in another*, (ii) *Any two hyperedges intersect in at most one node and* (iii) *The degree of any node is at most* $(k+1)$.

Proof. (i) As G is a k-tree, all of its base sets are of size k. So no base set of G is properly contained in another. By Theorem 8, no hyperedge of $H'(G)$ is contained in another.

(ii) Suppose E_1, E_2 are hyperedges of $H'(G)$ such that $|E_1 \cap E_2| > 1$. By definition, the *rch* is 'closed' under intersection, i.e., the intersection of any two hyperedges, if it is of size greater than one, is also a hyperedge of $H'(G)$. Thus $E_1 \cap E_2$ is a subhyperedge

of both E_1 and E_2. This violates property (i). Thus no two hyperedges intersect in more than one node.

(iii) The degree of a node q is the number of hyperedges which contain it. A hyperedge R contains the node q if the base set corresponding to R is contained in the maximal clique Q that corresponds to q (see Theorem 8). Thus, the degree of node q is equal to the number of base sets contained in Q. All maximal cliques of G are of size $(k+1)$ as it is a k-tree. Hence the maximum number of base sets Q can contain is

$$\binom{k+1}{k} = k+1.$$

Thus the degree of q is at most $(k+1)$. □

Lemma 9. *The* rch *of a k-tree is Berge-acyclic.*

Proof. Let C be a cycle in the *rch* $H'(G)$ of a k-tree G. $C = x_1E_1x_2E_2 \ldots x_rE_rx_1,\ r > 1$. By Lemma 8, the only possible cycle in $H'(G)$ is the one where nonadjacent edges are disjoint. But such a cycle can never be present in an *rch* because no tree can satisfy the *rch*. Thus $H'(G)$ is Berge-acyclic. □

The structural simplicity of k-trees is reflected in the simple tree-like structure of their *rch*.

Lemma 10. *The* rch *of a k-sep-chordal graph is Berge-acyclic.*

Proof. As all the base sets of a k-sep-chordal graph are of size k, properties (i) and (ii) given in Lemma 8 hold for k-sep-chordal graphs also. Note, however, that the bound on the degree of a node does not hold for k-sep-chordal graphs as they can have cliques of any size greater than $(k+1)$. Using the same arguments of the proof of Lemma 9, we can show that the *rch* of a k-sep-chordal graph is acyclic. □

This brings us to the following question: What is the largest subclass of chordal graphs that has the acyclic *rch* property? We call this class *arch*-chordal graphs (for **a**cyclic **rch** chordal graphs).

Theorem 13. *A chordal graph is an arch-chordal graph if and only if no two base sets are properly contained in each other.*

Proof. (If) The *rch* of a chordal graph G in which there is no proper containment between any two base sets satisfies the conditions (i) and (ii) given in the Lemma 8. Hence $H'(G)$ is acyclic.

(Only if) If B_1 and B_2 are any two base sets such that $B_1 \subset B_2$ then by Theorem 8, $R(B_2) \subset R(B_1)$ where $R(B_1)$ and $R(B_2)$ are the hyperedges corresponding to B_1 and B_2, respectively. It is easy to see that these edges give rise to a cycle of length two in $H'(G)$. □

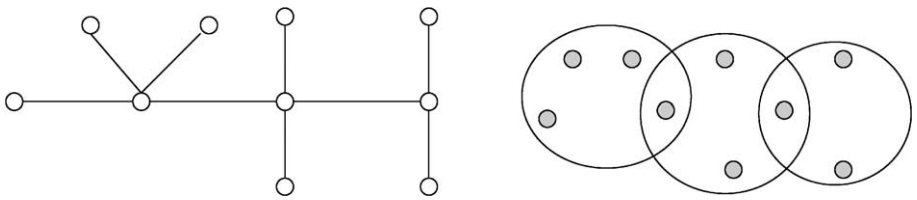

Fig. 6. The *rch* of a caterpillar.

The class of *arch*-chordal graphs is the largest subclass of chordal graphs that has the same simple clique structure as that of trees. It is also a natural generalization of the notion of k-trees and k-sep-chordal graphs.

Since the containment between a pair of base sets can be checked in $O(\omega(G))$ time, we have,

Theorem 14. *Recognition of arch-chordal graphs can be done in* $O(|V|^2\omega(G))$ *time.*

6.4. k-caterpillars

A tree T is called a *caterpillar* if the removal of all the leaves of T results in path [20]. Trotter et al. [21] have shown the following:

Lemma 11. *A tree is an interval graph if and only if it is a caterpillar.*

The *rch* of a caterpillar has a chain-like structure (see Fig. 6). It is clear from the algorithm Generate-Clique-Tree, that this structure ensures the existence of a clique tree which is a path. Now, a natural way of generalizing caterpillars is as follows: A k-tree G is called a k-caterpillar if its *rch* $H'(G)$ has distinct nodes $x_1, x_2, \ldots, x_r$ such that $x_1, E_1, x_2, E_2, \ldots, x_r, E_r$ is a path that uses all the hyperedges of $H'(G)$. Thus the *rch* of k-caterpillars also has a chain-like structure. The following is immediate.

Lemma 12. *A k-tree is an interval graph if and only if it is a k-caterpillar.*

The above definition of k-caterpillars is a good alternative to the definition proposed in [12].

Fig. 7 depicts the hierarchy of classes that emerges from the results presented in this section.

7. Conclusions

In this paper, we have introduced a new structure called the *reduced clique hypergraph* for chordal graphs and shown that all the clique trees of a chordal graph can

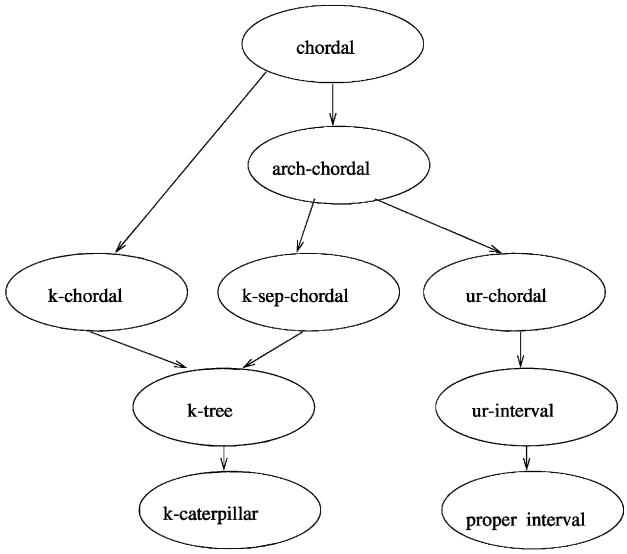

Fig. 7. A hierarchy of subclasses of chordal graphs.

be obtained from its *rch*. We have clearly brought out the connection between the *rch* structure and minimal vertex separators leading to efficient algorithms for counting and enumerating clique trees. We have also introduced several new subclasses of chordal graphs which are potentially useful in analyzing the complexity of hard problems on chordal graphs. Characterizing subclasses of chordal graphs in terms of their *rch* structure is an interesting course of future work.

Appendix

Here, we derive a formula for the number of clique trees of a connected chordal graph. Let $\mathscr{B} = \{B_1, B_2, \ldots, B_p\}$ be the set of base sets of a chordal graph G. As before, $R(B_i)$ denotes the hyperedge of $H'(G)$ that corresponds to the base set B_i. Recall that $d(B_i) = \mu(B_i) + 1$ is the number of components in $R(B_i)$. In the Algorithm Generate-Clique-Trees, we count the number of ways in which T can be obtained after computing $T_1, T_2, \ldots, T_{d(B_i)}$. Let s_{ij} be the number of nodes in T_j with the property that the maximal clique corresponding to the node properly contains B_i. Note that s_{ij} is also the size of the jth component in the hyperedge $R(B_i)$. We first fix the degrees of vertices in T'; let the degree of t_j be x_j. For each such tree T', we count the number of ways obtaining T as follows. For each edge (t_j, t_k) of T', we can choose an edge to be added to T in $s_{ij} * s_{ik}$ number of ways. The choice made at any edge of T' is independent of the choices made at other edges. Thus, the number of trees that can be obtained with T' is $\prod_{j=1}^{d(B_i)} (s_{ij})^{x_j}$. Hence the total number of trees obtained from

$T_1, T_2, \ldots, T_{d(B_i)}$, denoted by N_i, is given by the following:

$$N_i = \sum_{x_1+x_2+\cdots+x_{d(B_i)}=2d(B_i)-2} T(x_1, x_2, \ldots, x_{d(B_i)}) \prod_{j=1}^{d(B_i)} (s_{ij})^{x_j}.$$

Here, $T(x_1, x_2, \ldots, x_{d(B_i)})$ denotes the number of spanning trees on $\{t_1, t_2, \ldots, t_{d(B_i)}\}$ with t_j having degree x_j. By Prüfers proof of Cayley's formula (See [10]) for counting the number trees on n labeled vertices, we have

$$T(x_1, x_2, \ldots, x_{d(B_i)}) = \binom{d(B_i)-2}{x_1-1, x_2-1, \ldots, x_{d(B_i)}-1}.$$

Hence,

$$\begin{aligned} N_i &= \sum_{x_1+x_2+\cdots+x_{d(B_i)}=2d(B_i)-2} \binom{d(B_i)-2}{x_1-1, x_2-1, \ldots, x_{d(B_i)}-1} \prod_{j=1}^{d(B_i)} (s_{ij})^{x_j} \\ &= \prod_{j=1}^{d(B_i)} s_{ij} \sum_{x_1+x_2+\cdots+x_{d(B_i)}=d(B_i)-2} \binom{d(B_i)-2}{x_1, x_2, \ldots, x_{d(B_i)}} \prod_{j=1}^{d(B_i)} (s_{ij})^{x_j} \\ &= \prod_{j=1}^{d(B_i)} s_{ij} \left(\sum_{j=1}^{d(B_i)} s_{ij} \right)^{d(B_i)-2} \\ &= \prod_{j=1}^{d(B_i)} s_{ij} |R(B_i)|^{d(B_i)-2}. \end{aligned}$$

Now, it is clear from the Algorithm Generate-Clique-Trees, that the number of clique trees of a chordal graph G, denoted by $CT(G)$, is given by the following:

$$CT(G) = N_i \, CT(M_1) \, CT(M_2) \cdots CT(M_{d(B_i)}),$$

where M_j are the smaller chordal graphs that result after G is decomposed with respect to B_i.

As we have seen in the proof of Theorem 8, decomposing a chordal graph G with respect to a minimal base set B_i corresponds to removing a maximal hyperedge in the *rch* $H'(G)$. The resulting hypergraphs are the reduced clique hypergraphs of the smaller chordal graphs that arise after the decomposition. The set of base sets of G, except B_i, get partitioned among the chordal graphs that arise after the decomposition. During the recursive decomposition, every base set will at some stage become a minimal base set. Thus,

$$CT(G) = \prod_{i=1}^{|\mathscr{B}|} N_i = \prod_{i=1}^{|\mathscr{B}|} \left(\left(\prod_{j=1}^{d(B_i)} s_{ij} \right) |R(B_i)|^{d(B_i)-2} \right).$$

Hence, we have

Theorem 15. *The number of clique trees of a chordal graph G is given by*

$$CT(G) = \prod_{i=1}^{|\mathscr{B}|} \left(\left(\prod_{j=1}^{d(B_i)} s_{ij} \right) |R(B_i)|^{d(B_i)-2} \right).$$

We now address the problem of counting the number of clique trees of a chordal graph.

Recall that $d(B_i) = \mu(B_i) + 1$ denotes the number of connected components of $G - B_i$ that contain at least one vertex which is adjacent to all the vertices of B_i. s_{ij} denotes the number of maximal cliques that properly contain B_i in the jth component. Further, $|R(B_i)| = \sum_{j=1}^{d(B_i)} s_{ij}$. Thus, computing s_{ij} for all i is sufficient for computing the value of $CT(G)$.

To compute s_{ij} for B_i, we proceed as follows. Associate a counter, initialized to zero, with each vertex of G. For each vertex $x \in B_i$, increment the counter of all the vertices that are adjacent to x. Repeat this for all vertices of B_i. Next, mark all those vertices of G whose counter value equals $|B_i|$ as red vertices. Note that each red vertex is adjacent to all the vertices of B_i. Now, determine the connected components of $G - B_i$. In each component that contains at least one red vertex, take the subgraph induced by the red vertices and compute the size of its maximum independent set. It is easy to see that this number equals the number of maximal cliques of the component that properly contain B_i. As the induced subgraphs mentioned above are all chordal, computing the size of the maximum independent set for all of the components takes $O(|V| + |E|)$ time [8]. Determining the red vertices takes $O(|V||B_i|)$ time. Thus computing s_{ij} for a base set B_i takes $O(|V||B_i| + |V| + |E|) = O(\omega(G)|V| + |E|)$ time and computing s_{ij} for all i takes $O(|\mathscr{B}|(\omega(G)|V| + |E|))$ time.

The time required to compute $CT(G)$ is the total time required to compute the base sets and s_{ij} for all i. Hence,

Theorem 16. *The number of clique trees of a chordal graph G can be computed in $O(|\mathscr{B}|(\omega(G)|V| + |E|))$ time, where $\mathscr{B}$ is its set of base sets.*

References

[1] P.A. Bernstein, N. Goodman, Power of natural semijoins, SIAM J. Comput. 10 (1981) 751–771.
[2] K.S. Booth, J.H. Johnson, Dominating sets in chordal graphs, SIAM J. Comput. 11 (1) (1982) 191–199.
[3] P. Buneman, A characterization of rigid circuit graphs, Discrete Math. 9 (1974) 205–212.
[4] G.A. Dirac, On rigid circuit graphs, Abh. Math. Sem. Univ. Hamburg 25 (1961) 71–76.
[5] D.R. Fulkerson, O.A. Gross, Incidence matrices and interval graphs, Pacific J. of Math. 15 (1965) 835–855.
[6] F. Gavril, The intersection graphs of subtrees in trees are exactly the chordal graphs, J. Combin. Theory Ser. B 116 (1974) 47–56.
[7] F. Gavril, Generating the maximum spanning trees of a weighted graph, J. Algorithms 8 (1987) 592–597.
[8] M.C. Golumbic, Algorithmic Graph Theory and Perfect Graphs, Academic Press, New York, 1980.

[9] C. Ho, R.C.T. Lee, Counting clique trees and computing perfect elimination schemes in parallel, Inform. Process. Lett. 31 (1989) 61–68.
[10] J.W. Moon, Various proofs of Cayley's formula for counting trees, in: F. Harary (Ed.), A Seminar on Graph Theory, Holt, Rinehart and Winston, New York, 1967, pp. 70–78.
[11] J. Naor, M. Naor, A.A. Schaffer, Fast parallel algorithms for chordal graphs, ACM Symposium on Theory of Computing, 1987, pp. 355–363.
[12] A. Proskurowski, Separating subgraphs in k-trees: cables and caterpillars, Discrete Math. 49 (1984) 275–285.
[13] Y. Shibata, On the tree representation of chordal graphs, J. Graph Theory 12 (1988) 421–428.
[14] P. Sreenivasa Kumar, Algorithmic and structural results on chordal graphs, Ph.D. Thesis, Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India, September 1990.
[15] P. Sreenivasa Kumar, C.E. Veni Madhavan, Minimal vertex separators of chordal graphs, Discrete Appl. Math. 89 (1998) 155–168.
[16] P. Sreenivasa Kumar, C.E. Veni Madhavan, 2-separator Chordal graphs, Proceedings of the Third National Symposium on Theoretical Computer Science, Kharagpur, June 1993, pp. 37–52.
[17] R.E. Tarjan, M. Yannakakis, Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs and selectively reduce acyclic hypergraphs, SIAM J. Comput. 13 (3) (1984) 565–579.
[18] N.Ch. Veeraraghavulu, P. Sreenivasa Kumar, C.E. Veni Madhavan, A linear-time algorithm for isomorphism of a subclass of chordal graphs, Inform. Process. Lett. 44 (1992) 45–50.
[19] F.S. Roberts, ed., Applications of Combinatorics and Graph Theory to the Biological and Social Sciences, Vol. 17, The IMA volumes in Mathematics and Applications, Springer, 1989.
[20] F. Harary and A.J. Schwenk, Trees with Hamiltonian squares, Mathematica, 18 (1971) 138–140.
[21] W.T. Trotter, Jr. and F. Harary, On double and multiple interval graphs, J. Graph Theory 3 (1979) 205–211.